Chapter 15
Swarm Robotics

Factories use multiple robots to achieve goals such as painting and welding a car (Fig. 1.3). The use of multiple robots shortens manufacturing time by performing different tasks simultaneously such as welding parts on both sides of a car. These tasks are typically designed to be independent with no close collaboration between the robots. Later, robots, especially mobile robots, were designed to collaborate with each other *directly* to perform multiple actions simultaneously in different places.

Here are some examples of tasks that require multiple robots to collaborate:

- Manipulating large structural elements in buildings, as well as in environments that are hard for humans to access such as in space or underwater.
- Performing a task through the collaboration of different types of robots: in a large-scale disaster, a flying drone can locate areas where victims are likely to be found, while a tracked robot on the ground searches these areas, focusing on places that may be hidden from aerial observation by trees or rubble.
- Performing simultaneous measurements in different locations: measuring sound disturbances in different parts of a building, or monitoring pollution after an industrial accident.

What is common to these situations is that multiple robots participate in performing a task and they need to coordinate with each other because they are acting on the same physical object even though they are not next to each other in the environment.

Collaboration among robots can also be used to speed up the execution of a task by having several robots perform the task in parallel. Consider measuring pollution over a large area: a single robot can roam the entire area (like a robotic vacuum cleaner in an apartment), but the task will be accomplished much faster if multiple robots divide up the areas to be covered among themselves.

M. Ben-Ari and F. Mondada, *Elements of Robotics*,
https://doi.org/10.1007/978-3-319-62533-1_15

15.1 Approaches to Implementing Robot Collaboration

There are two main approaches to the design of systems composed of multiple robots. The first is a *centralized system*, where a central component (one of the robots or an external computer) coordinates all the robots and their tasks. The advantage of a centralized system is that it is relatively simple to implement. The main disadvantage is that it is difficult to expand because adding more robots adds processing load to the central station where all the intelligence is concentrated. In a centralized system, the robots themselves can be "dumb," but most robots have significant computing power that is not well utilized in this architecture. Another serious disadvantage of a centralized system is that the central component is a single point of failure. If it stops working the entire systems fails. In critical environments it is unacceptable to employ a system that is not robust under the failure of a single component.

If we look to the animal world, we see that many activities are *distributed*, that is, independent individuals work together to achieve common goals of the entire population. Ants optimize their path to food sources not by depending on one ant to dispatch others to search and then to process the information returned, but by a distributed effort of the entire ant colony. Individual ants mark the ground with pheromones that are sensed by the other ants. If a few ants are eaten by a predator, the rest of the colony survives, as does the knowledge embodied in the locations of the pheromones. The efficiency and robustness of this approach was demonstrated in the algorithms in Chap. 7.

Swarm robotics is a distributed approach to robotics that tries to coordinate behavior by copying mechanisms inspired by the behavior of social animals. These mechanisms, often local and simple, allow a group to achieve global performance that could not be achieved by an individual on its own. Distributed systems have the following advantages:

- They are robust. Losing one out of ten robots only reduces the performance of the system by about ten percent instead of causing failure of the entire system.
- They are flexible and scalable. The number of robots can be adapted to the task. If there are ten robots in the system but five are sufficient to perform the task, the other five can be assigned to other tasks, while if ten robots cannot perform the task efficiently, another ten can easily be added.

These advantages come at the cost of the effort required to design and implement coordination among the robots. In swarm robotics, as well as in nature, there are relatively simple coordination mechanisms that make distributed systems feasible.

This chapter presents two approaches to coordination in swarm robotics:

- Information-based coordination (Sect. 15.2), where the interaction is in the form of communications between robots. This can be either directly by explicitly passing electronic messages or indirectly by placing messages in the environment.
- Physical coordination (Sect. 15.3), where individual robots interact at the mechanical level, either directly by exerting forces on each other or indirectly by manipulating a common object.

15.2 Coordination by Local Exchange of Information

Communications can be either global or local. Suppose that you receive a call from your friends and they inform you: "We see an ice-cream store on the left." This *global* information is useless unless they give you their current location. However, if you are walking side-by-side with them and one says: "I see an ice-cream store on the left," from this *local* information you immediately know the approximate location of the store and can easily locate it visually.

Inspired by nature, swarm robotics uses local communications within a distributed architecture. A few zebras on the outer edges of a herd look for predators and signal the others by sound or movement. Herding is a successful survival strategy for animals because local communications enables large numbers of animals to flee immediately upon detection of a predator by a small numbers of alert watchers.

15.2.1 Direct Communications

Direct local exchange of information is achieved when a friend talks to you. Animals don't talk but they do use sound as well as movement and physical contact to achieve direct local exchange of information. Robots implement direct local communications electronically (such as local WiFi or Bluetooth), or by transmitting and receiving light or sound. Alternatively, they can use a camera to detect changes in another robot such as turning on a light.

Local communications can be either *directional* or *non-directional*. Radio communications such as Bluetooth is local (just a few meters) and generally the receiving robot does not attempt to determine the direction to the transmitting robot. Local directional communications can be implemented using a light source as the transmitter and a narrow-aperture detector or a camera as the receiver.

15.2.2 Indirect Communications

Indirect local communications refers to communications through a medium that can store a transmitted message for later access. The most familiar example is mail, either email or regular mail, where the transmitter composes the message and sends it, but the message remains at the server or the post office until it is delivered to the receiver who, in turn, may not access it immediately. Indirect communications in animals is called *stigmergy*; animals leave messages by depositing chemical substances that other animals can sense. We mentioned the use of pheromones by ants; another example is the use of urine by a dog to mark its territory.

Chemical messages are hard to implement in robots, but robots can leave optical markings on the ground as we did in the algorithm simulating a colony of ants

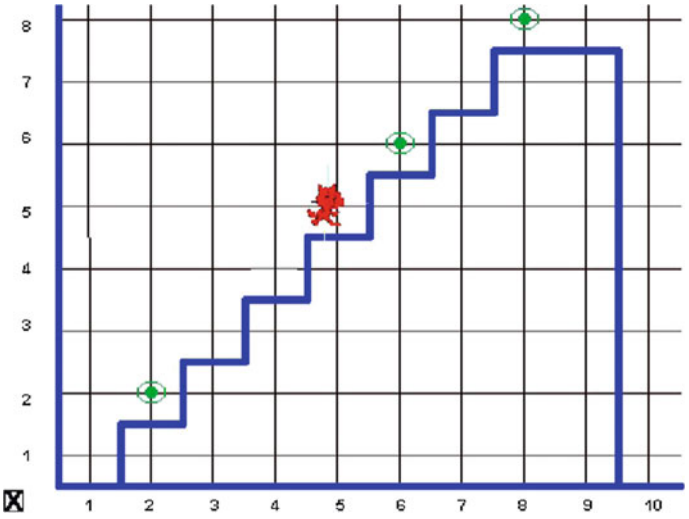

Fig. 15.1 Karel the Robot implemented in Scratch; the *green dots* are the beepers

searching for a food source (Sect. 7.3). There only one robot was used, but the algorithms could be easily implemented with multiple robots because it is the marking that is important, not the identity of the robot that created the marking.

Indirect communications can also be implemented by placing or manipulating objects in the environment. Robotic vacuum cleaners use *beacons* that can be placed at the entrance of rooms that the robot should avoid. It is possible to conceive of beacons that record when a room has already been cleaned and this information is (indirectly) communicated to other vacuum cleaners.

Karel the Robot is an environment used to teach programming. Commands move a virtual robot around a grid on a computer screen and the robot can deposit and sense "beepers" place on squares of the grid (Fig. 15.1).[1]

Indirect communications when combined with manipulation can generate interesting patterns. Figure 15.2a shows an environment filled with small objects and five mobile robots equipped with grippers. The robots follow a simple set of rules:

- If the robot finds an isolated object it picks the object up;
- If the robot finds an isolated object but already holds one, it puts the new object down next to the one that was found;
- The robot avoids walls and groups of several objects.

It seems that these rules will cause the robots to eventually place all the objects in groups of two, but this does not happen as shown in Fig. 15.2b. The reason is that a group of objects *seen from the side* can look like an isolated object, so an additional

[1]The image is taken from the first author's implementation of Karel the Robot in Scratch (https://scratch.mit.edu/studios/520857).

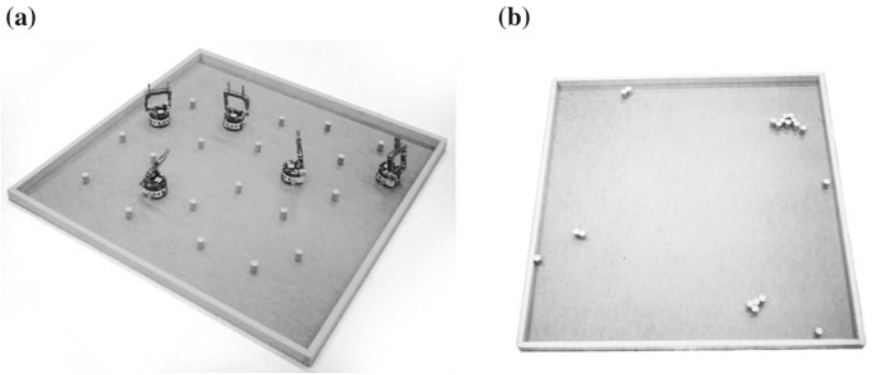

Fig. 15.2 **a** Gripper robots in an environment filled with small objects. **b** The objects have been collected into groups

object is placed in the group. The result is that large groups of objects are assembled purely by indirect communication.

15.2.3 The BeeClust Algorithm

The BeeClust algorithm is a swarm algorithm inspired by the behavior of bees. It uses a distributed architecture and local communications to generate a global result. The BeeClust algorithm is based on the way that very young bees cluster around locations of optimal temperature in the darkness of their nest. They measure local temperatures and detect collisions with other bees. The algorithm can be used by a swarm of robots to locate pollution; instead of measuring temperature, each robot measures some physical quantity that indicates levels of pollution. In time, the robots will come together in groups at locations of high levels of pollution.

Figure 15.3 shows a state machine implementing the algorithm. The robot moves randomly until it hits another robot, at which point it measures the temperature at the location of the collision. It waits at this location for a period of time proportional to the temperature that it found and then returns to moving randomly. When the robot moves it avoids obstacles such as walls. The algorithm uses what is perhaps the simplest form of communications: detecting a collision with another robot. The localized nature of the communications is essential for the correct functioning of the algorithm.

Initially, the robots will collide at random places, but those that are in locations with higher temperatures will remain there for longer periods of time which in turn causes additional collisions. In the long term, most of the robots will form a cluster in the area with the highest temperatures. They collide with each other frequently since being in a crowd increases the number of collisions. Of course this mechanism

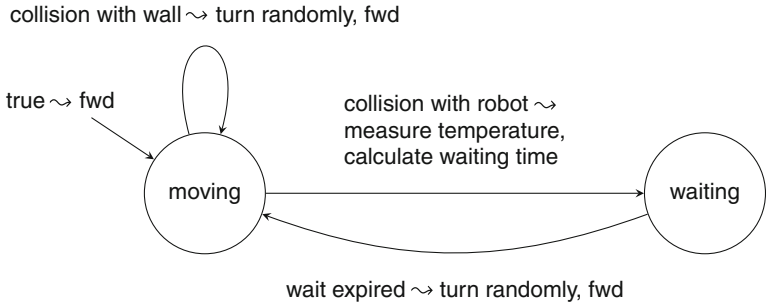

Fig. 15.3 BeeClust algorithm

can only work with a large number of robots that generate many collisions leading to numerous measurements and clustering.

15.2.4 The ASSISIbf Implementation of BeeClust

Within the ASSISIbf project, researchers at the University of Graz used Thymio robots to implement the BeeClust algorithm as part of research into the behavior of bees in a nest. The algorithm simulates a group of young bees in a cold circular arena, where two virtual sources of heat are placed on the right and on the left of the arena. Initially, only one of these sources of heat is active. The heat sources are simulated by two clusters of three robots at the edges of the arena (Fig. 15.4a).

The three heat-emitting robots on the left side transmit their temperature. Bee-robots in their vicinity detect this signal and stop for a period of time proportional to the temperature. During their stay in this area they also transmit a temperature signal. Furthermore, if the heat-emitting robots detect nearby robots they increase their heat and transmit the new temperature. Figure 15.4 shows the evolution of the behavior of the robots: (a) the initial state when the robots start moving and only the left temperature source is on; (b) the robots start clustering around the left source; (c) the highest level of clustering occurs. The bee-robots can leave the cluster, explore the environment and return to the cluster as shown in (d). This is caused by the random nature of the algorithm and is essential to prevent the system from being trapped in local minima. Figure 15.4e shows the moment when the right source is also switched on. After about 21 min the robots form two smaller clusters (f).

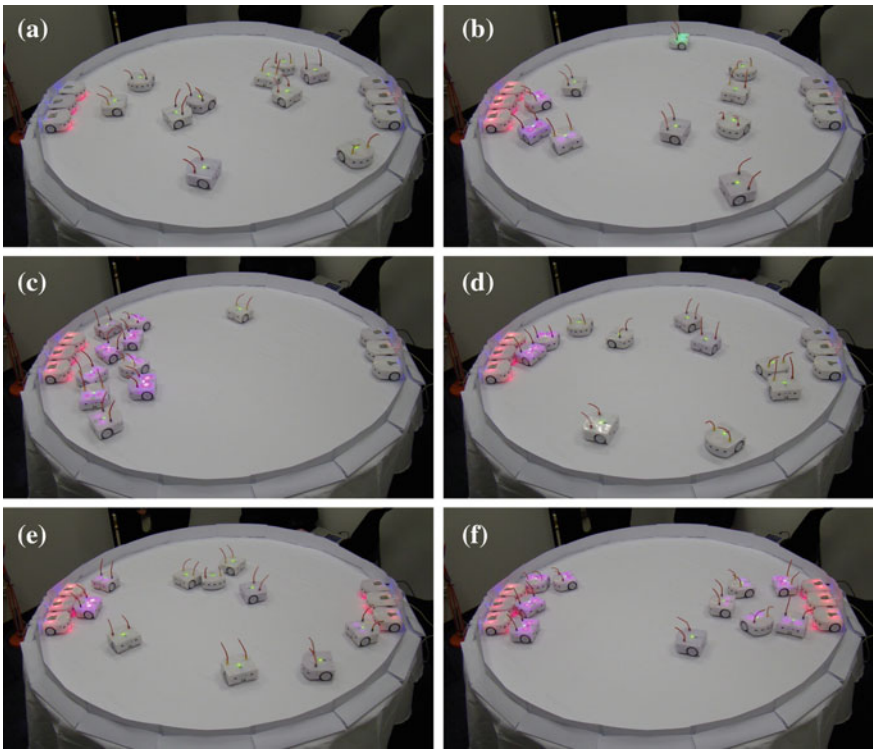

Fig. 15.4 BeeClust implementation. The photographs were taken at the following times (min:sec) from the beginning of the experiment: **a** 1:40, **b** 2:30, **c** 9:40, **d** 12:40, **e** 13:20, **f** 21:10

Activity 15.1: BeeClust algorithm

- Implement the BeeClust algorithm to cause a group of robots to cluster at the brightest location in a room.
- Use three sensors: a light sensor to measure the ambient light, a sensor to detect other robots and a sensor to detect the limits of the arena.
- Solution 1: Use proximity sensors to detect other robots and ground sensors to detect a line defining the limits of the arena.
- Solution 2: Define the limits of the arena with a wall and use robot-to-robot communications to distinguish between robots and the wall. To avoid confusion do not use the same sensor to detect the other robots and the wall.

15.3 Swarm Robotics Based on Physical Interactions

In Sect. 7.2 we looked at a typical example of efficient swarm behavior: a colony of ants finding a path from their nest to a source of food. That example used indirect communications in the form of pheromones deposited on the ground. In this section we look at another form of swarm behavior that is mediated by physical interactions. We start with ants collaborating on the task of pulling a stick from the ground and a robotic version of this task. This is followed by a discussion of how forces exerted by several robots can be combined, demonstrated by a simple but clever algorithm called occlusion-based collective pushing.

15.3.1 Collaborating on a Physical Task

Figure 15.5a shows two ants extracting a stick from the ground for use in building a nest. The stick is embedded so deeply that one ant cannot extract it by itself. We want to design a robotic system to accomplish this task (Fig. 15.5b). Each robot searches randomly until it finds a stick. It then pulls on the stick as hard as possible. If it successfully extracts the stick, it takes it back to a nest; otherwise, since it has only partially extracted the stick, it waits until it feels that another robot is pulling harder and releases the stick. If no robot comes to its help for a period of time, the robot releases its hold and returns to a random search. This ensures that if there are more sticks than robots, the system won't deadlock with each robot trying to extract one stick and waiting indefinitely.

The finite state machine for this algorithm is shown in Fig. 15.6. Although this behavior is simple and local, when applied by two robots it results in the extraction of the stick from the ground. The robot on the right in Fig. 15.5b pulls part of the stick as far as possible out of the ground using the maximum movement of its arm. When it detects that another robot has found the same stick and starts pulling, the first robot releases the stick to allow the second robot to extract it. By combining the

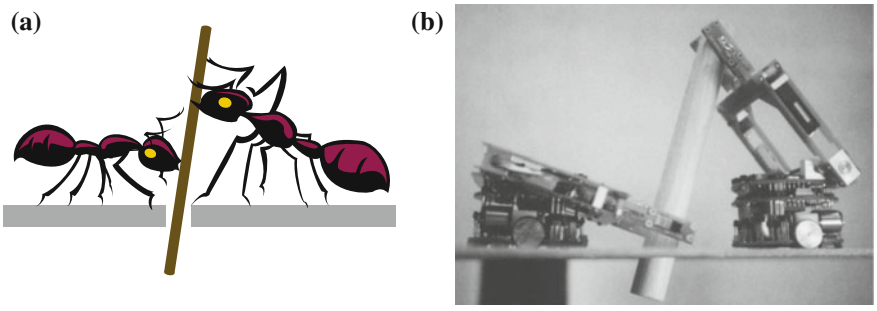

Fig. 15.5 **a** Ants pulling a stick from the ground. **b** Robots pulling a stick from the ground

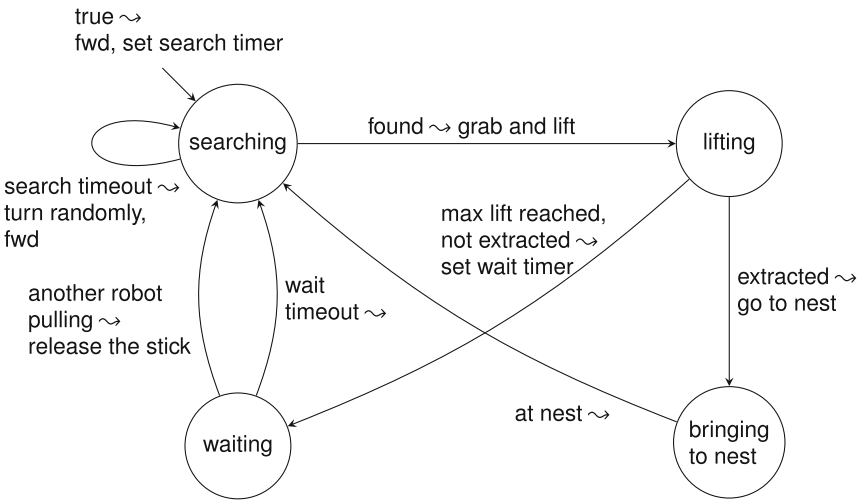

Fig. 15.6 Algorithm for distributed stick pulling

physical capability of two robots with a simple specification of behavior, we achieve a result that neither of the robots could achieve alone.

15.3.2 Combining the Forces of Multiple Robots

Figure 15.7 shows a differential-drive robot moving backwards (from left to right). It exerts a force F_r that can be used to pull an object. Figure 15.8 shows two robots connected together so that they exert a force F_{total} when moving from left to right.

What is the relationship between F_r and F_{total}? There are three possibilities:

- $F_{total} < 2F_r$: The connected robots lose efficiency because the force they exert is less than that exerted by the two robots pulling separately.
- $F_{total} = 2F_r$: The connected robots achieve the same efficiency as two separate robots.
- $F_{total} > 2F_r$: The connected robots are more efficient that two separate ones.

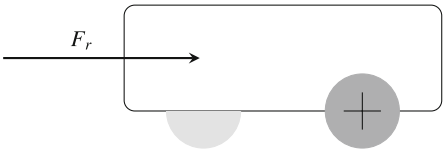

Fig. 15.7 One robot pulling with a given force F_r

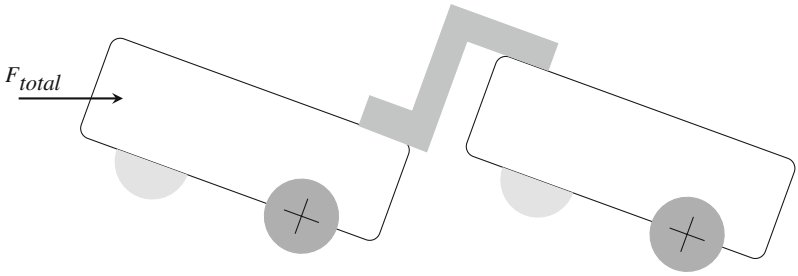

Fig. 15.8 Two connected robots pulling with a given force F_{total}

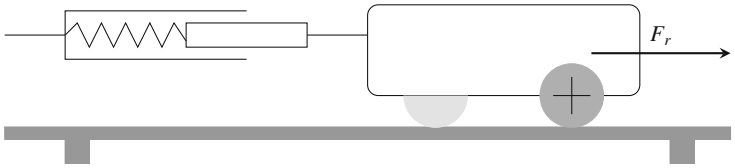

Fig. 15.9 Measuring the force with a dynamometer

Robots can achieve $F_{total} > 2F_r$, where the total force is greater than the sum of the forces exerted by the individual robots, because in some mechanical configurations the connected robots are more stable since their center of gravity better placed.

Activity 15.2: Pulling force by several robots.

- Connect two robots and check whether they exert a force that is less than, the same or greater than twice that exerted by a single robot. You can connect the robots with a rigid connection as in Fig. 15.8 or with a flexible connection using string.
- Measure F_r, the force exerted by a single robot, and then measure F_{total} the force exerted by the connected robots.
- A *dynamometer* (Fig. 15.9) is the best instrument for measuring forces. If you do not have one available, you can use a cable, a pulley and weights (or a scale) as shown in Fig. 15.10.
- Change the orientation of the robots: pulling forwards instead of backwards. Does this change the result?
- Experiment on different surfaces (hard ground, carpet, paper) and determine the effect of the surface on the resulting force.

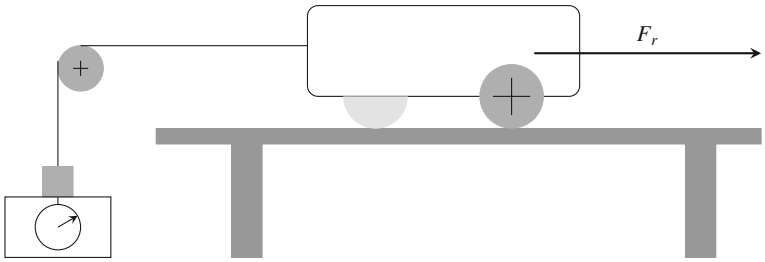

Fig. 15.10 Measuring the force with a weight and a scale

15.3.3 Occlusion-Based Collective Pushing

Let us consider another example of combining the force of multiple robots. Instead of a physical link as in Fig. 15.8, we use many simple robots to push an object, imitating a group of ants. Again, the advantages of a distributed system are flexibility—engaging just the resources needed for the task—and robustness—if one robot fails, the object might move a bit slower but complete failure of the task does not occur. These advantages come at the cost of additional resources and the complexity of implementing coordination among the robots.

Figure 15.11 shows a group of small robots pushing a large object represented by the circle towards a goal. One approach would be to determine the direction to the

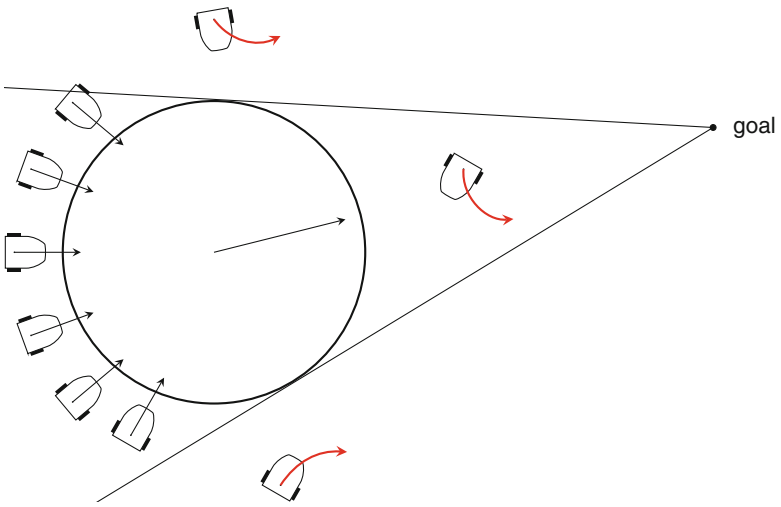

Fig. 15.11 Occlusion based coordination: straight *black arrows* for the robots pushing the object and curved *red arrows* for the robots searching for an occluded position

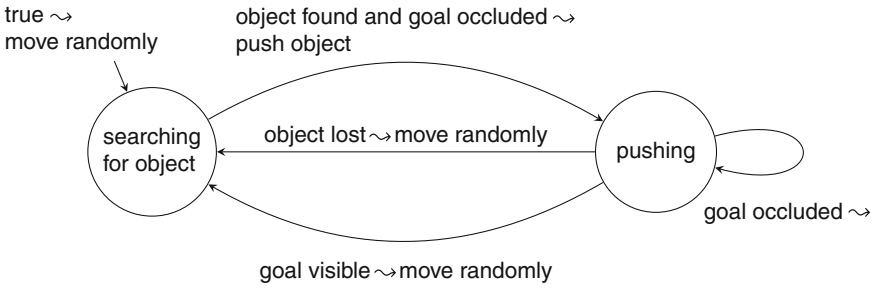

Fig. 15.12 Algorithm for occlusion-based coordination

goal, to share this information among all the robots and to have each robot compute the direction it should push. This is not as simple as it may appear since a simple robot might not be able to determine its absolute position and heading.

A swarm robotics approach is called *occlusion-based pushing*, which assumes that the robots have only local knowledge, not global knowledge of what the other robots are doing. The robots are able to determine whether they can detect the goal or not. For example, a bright light could be mounted on the goal and the robots equipped with a light sensor.

Figure 15.12 shows the finite state machine for this algorithm. The robots search for the object; when they find it they place themselves perpendicular to the surface and push (straight black arrows). This could be implemented using a touch sensor or a proximity sensor. The robots continue to push as long as they *do not* detect the goal. If they do detect the goal, they stop pushing, move away (curved red arrows) and commence a new search for an occluded position where they resume pushing. The result is that the vector sum of the forces exerted by the robots is in a direction that moves the object toward the goal. The occlusion algorithm leads to the task being performed without a central control unit and even without inter-robot communications.

Activity 15.3: Total force

- Consider the configuration shown in Fig. 15.13. Robot 1 is pushing an object at a 45° angle with force f_1, robot 2 is pushing horizontally with force f_2 and robot 3 is pushing vertically with force f_3. Show that the f_{total}, the total force on the object, has magnitude:

$$\sqrt{\left(f_2 + \frac{f_1}{\sqrt{2}}\right)^2 + \left(f_3 - \frac{f_1}{\sqrt{2}}\right)^2}.$$

  in the direction:

$$\alpha = \tan^{-1} \frac{2f_3 - \sqrt{2}f_1}{2f_2 + \sqrt{2}f_1}.$$

- Compute the magnitude and direction of f_{total} for various values of the individual forces. If $f_1 = f_2 = f_3 = 1$, the magnitude is $\sqrt{3}$ and the direction is 9.7°. If $f_1 = f_2 = 1$, what must f_3 be so that $\alpha = 45°$?
- Use three robots to implement this configuration and check that the object moves in the computed direction.

Activity 15.4: Occlusion-based pushing

- Place three robots around an object and a place a goal some distance away from the robot. Implement a mechanism for the robots to determine the direction to the goal, for example, attach a light to the goal or place the goal at the lowest point of an inclined surface.
- Implement a mechanism that enables the robots to distinguish between the object and the boundary of the surface on which they move. For example, use a black tape on the surface for its boundary and detect the object using proximity sensors.
- Implement a mechanism so that robots do not push each other. One method would be to use a color sensor and attach colored tape to the robots.
- Implement the occlusion-based pushing algorithm.
- Discuss whether occlusion-based pushing could be used in three dimensions, for example, underwater robots pushing an object.

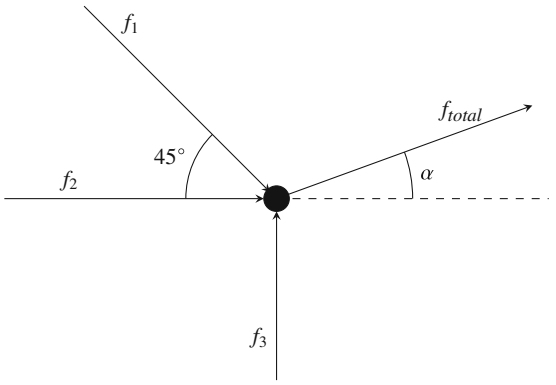

Fig. 15.13 The total force from three robots

15.4 Summary

Swarm robotics uses multiple robots in a distributed architecture to perform a task. With a distributed architecture, the system is robust to the failure of individual robots and flexible in its ability to add or remove robots as the scale of the task changes. A distributed architecture can perform tasks that individual robots cannot, as we saw in the examples of robots combining forces. Finally, multiple robots can act simultaneously at different locations that are far from each other. These advantages come at a price: the increased cost of the multiple robots, the complexity of the coordination mechanisms that must be implemented, and in some cases the loss of performance due to the overlap between the action of different robots.

15.5 Further Reading

An overview of collective robotics is given in [5] and the collection [8] focuses on swarm robotics. For the specific projects presented in this chapters see:

- Karel the Robot [6].
- BeeClust [1, 7].
- ASSISIbf [9] and http://assisi-project.eu.
- Physical interaction (pulling a stick) [4].
- Occlusion-based transport [2, 3].

References

1. Bodi, M., Thenius, R., Szopek, M., Schmickl, T., Crailsheim, K.: Interaction of robot swarms using the honeybee-inspired control algorithm beeclust. Math. Comput. Model. Dyn. Syst. **18**(1), 87–100 (2012)
2. Chen, J., Gauci, M., Groß, R.: A strategy for transporting tall objects with a swarm of miniature mobile robots. In: IEEE International Conference on Robotics and Automation, pp. 863–869 (2013)
3. Chen, J., Gauci, M., Li, W., Kolling, A., Groß, R.: Occlusion-based cooperative transport with a swarm of miniature mobile robots. IEEE Trans. Robot. **31**(2), 307–321 (2015)
4. Ijspeert, A.J., Martinoli, A., Billard, A., Gambardella, L.M.: Collaboration through the exploitation of local interactions in autonomous collective robotics: the stick pulling experiment. Auton. Robots **11**(2), 149–171 (2001)
5. Kernbach, S.: Handbook of Collective Robotics: Fundamentals and Challenges. CRC Press, Boca Raton (2013)
6. Pattis, R.E., Roberts, J., Stehlik, M.: Karel the Robot: A Gentle Introduction to the Art of Programming. Wiley, New York (1995)
7. Şahin, E., Spears, W.M. (eds.): Swarm robotics: from sources of inspiration to domains of application. Swarm Robotics: SAB 2004 International Workshop, pp. 10–20. Springer, Berlin (2005)

8. Schmickl, T., Thenius, R., Moeslinger, C., Radspieler, G., Kernbach, S., Szymanski, M., Crailsheim, K.: Get in touch: cooperative decision making based on robot-to-robot collisions. Auton. Agents Multi-Agent Syst. **18**(1), 133–155 (2009)
9. Schmickl, T., Bogdan, S., Correia, L., Kernbach, S., Mondada, F., Bodi, M., Gribovskiy, A., Hahshold, S., Miklic, D., Szopek, M., et al.: Assisi: mixing animals with robots in a hybrid society. In: Conference on Biomimetic and Biohybrid Systems, pp. 441–443. Springer, Berlin (2013)